

S/N 10/732,929

**PATENT**

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

Applicant: James R. Kohn  
Serial No.: 10/643,574  
Filed: August 18, 2003  
Title: INDIRECTLY ADDRESSED VECTOR LOAD-OPERATE-STORE  
METHOD AND APPARATUS  
Examiner: Robert E. Fennema  
Group Art Unit: 2183  
Docket: 1376.730US1

DECLARATION UNDER 37 C.F.R. § 1.131

Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

This declaration is submitted under 37 C.F.R. § 1.131 in response to the rejection of U.S. Patent Application Serial Number 10/643,574 (the "Application"), assigned to Cray Inc. to establish the inapplicability of using the reference "Cray Assembly Language (CAL) for Cray X1™ Systems Reference Manual," published June, 2003 (the Cray Manual), to reject the claims of the instant application under 35 USC § 103(a). U.S. Patent Application Serial No. 10/643,574 was filed August 18, 2003.

J. James R. Kohn, do hereby declare:

1. I am currently an employee of Cray Inc, the assignee of the Application and publisher of the Cray Manual, and have been an employee since at least as early as December 9, 2002.
2. I am the inventor of the claims of the Application.
3. The subject matter claimed in the Application was invented prior to June, 2003, the publication date of the Cray Manual.
4. The enclosed document shows a copy of an original e-mail dated December 9, 2002 that I (Jim Kohn) authored. The e-mail includes code that I created which implements the invention of the Application.
5. The subject matter claimed in the Application was invented in the United States.

6. I further declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true, and further that these statements are made with the knowledge that willful false statements and the like are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of this application or any patent issuing thereon.

Date:

July 6, 2007 James R. Kohn  
James R. Kohn

Paul Urbanski

---

----- Original Message -----

Subject: Re: Vector update vfunction  
Date: Mon, 09 Dec 2002 13:49:36 -0600  
From: Jim Kohn <jkohn@cray.com>  
To: Terry Greyzck <tdg@cray.com>, Vince Graziano <vfg@cray.com>, Charlie Carroll <charliec@cray.com>, Lyle Williams <lew@cray.com>  
References: <200211071804.gA7I4gSH11272221@mint.us.cray.com> <3DEE5E8B.FF08C906@cray.com>

All,

We need to start looking at what is required to get the new vector update algorithm into the compiler. This algorithm is multistreamable and on the HMG Tabletoy benchmark can achieve a 4x speedup over the current non-streamable algorithm. We are close to the customer's expectation. We would need to be able to demonstrate this in a compiler by June 2003.

As a start, I've attached the inline version of the "indexed partial reduction" code that I used to achieve this speedup. I also show the bracketing code where the ordered msync masks are created and used to order the final update of memory.

A possibility to consider is the inlining of vfunction code with a "suppress" of the scratch A, S, and V regs so that these are available to the vfunction code. The inlining of EXP has already been requested. The "indexed partial reduction" is a second instance. There may be later vfunction algorithms that we may want to consider inlining as well such as pack, expand, search, etc. especially multistreamed versions of these.

I will be out of the office for the rest of today and tomorrow attending a funeral. We can discuss this further on Wednesday morning or Thursday.

Jim

---

```
*   HMG Tabletoy update:  table[xdata.index[i]] += xdata.value[i]
*
*   Registers computed or loaded during RHS processing of update...
*
*       v2      [a27,2],m0          ;IX = xdata.index[*]
*       v0      cidx(a11,m0)       ;IOTA
*       m1      m0|m0             ;input mask
*       v1      [a28,2],m0          ;Y = xdata.value[*]
*
*   Generate ordered msync wait,send masks
*
*   A10 = Remaining tripcount (after this pass)
*   A11 = 1
*   A22 = SSP#
*   A26 = SSP's array offset
*
*       a24    a22^3            ;=0 iff P3
*       a25    a0<a26            ;=0 iff P0 and 1st iter, else 1
*       a24    a10|a24           ;=0 iff P3 and last iteration
*       a21    a22-1
```

```

a26 a0<a24           ;=0 iff P3 and no more iters, else 1
a23 a22+1
a21 a21&3           ;restrict shift counts to be 0..3
a23 a23&3
a22 a11<<a22       ;self-mask
a21 a25<<a21       ;mask for SSP to wait on
a23 a26<<a23       ;mask for SSP to send
a21 a21|a22          ;wait mask
a22 a22|a23          ;send mask

* Inlined "indexed partial reduction" algorithm:  Y',M1 = reduce(Y, IX),M1
*
* Y' will contain Y or sum reduced values of Y for duplicate IX values;
* M1 will contain an update mask where IX values are unique and also where
* the Y' elements that need to be added into the update (LHS) vector.
*
* Input:
*   v0 = IOTA vector (0,1,2,...,63)
*   v1 = Y vector
*   v2 = IX vector
*   m1 = Input mask
*   v1 = #elements in v0, v1, v2
*
* Output:
*   v1 = Y' vector
*   v2 = IX vector
*   m1 = Output mask of unique IX values

```

```

CNFXSZ      =      16384           ;Size of scratch conflict analysis space

s4  CNFXSZ-1
a29  v1
a45  CNFXSZ*8-8
v5  v2&s4,m0           ;Conflict index set masked from ix
m4  fill(a29)
m3  m1&m4
a20  CNFXSZ*8
a45  a63-a45
s28  8
a63  a63-a20          ;Allocate private stack space

v6  v2<<s28,m0
v4  v6|v0,m0
a27  last(m4)          ;(ix<<8) to make room for IOTA
                       ;(ix<<8)|IOTA
                       ;last valid element#

```

```

cnfxloop =  *           ;"False positive" conflict loop
  [a45,v5] v4,m3,ord  ;Scatter (ix<<8)|IOTA (to scratch array)
  s27  x'00ff:d
  lsync v,v
  v6  [a45,v5],m3     ;Gather (ix<<8)|IOTA'

  v7  +v6>>s28,m3   ;Extract ix'
  m2  v7==v2,m3       ;M2 excludes ix's mapping to same CNFX

  v9  v6&s27,m3
  m4  v9!=v0,m2
  m3  ~m2&m3          ;Element #s of y sums
                       ;Conflict map
                       ;Map of remaining ix values

  a6  1
  a29  pop(m4)         ;Conflict trip count (tc)

  v7  cmprss(v9,m4)   ;IOTA's that conflicts map to
  a26  pop(m3)          ;>0 if ix's mapped to same CNFX
  m1  ~m4&m1           ;Exclude conflicts in final M1

  a1  v7,0
  a8  a6<a29           ;1st iota into which to sum (iota)
                       ;=1 if tc > 1

```

```

v7,a29      a27      ;Store safe y sum index at end
a6      a0<a29      ;=1 if tc > 0
a7      a6+a8      ;=2 if tc > 1, else tc

a2      v7,a6      ;2nd iota into which to sum (iota2)
a3      v7,a7      ;3rd iota into which to sum (iota3)

v8      cmprss(v1,m4)      ;y values to add into y sums
bz      a29,noconflict      ;If no conflicts exist

a11     v8,0      ;Get 1st 3 y values (y1,y2,y3)
v8,a29      s0      ;Store 0 for conflict summing at end
a12     v8,a6
s3      v8,a7

$REPEAT
a5      a7<a29      ;Repeat 3 update fixes per iteration
s5      v1,a1      ;=1 if >=0 more conflicts (another iter)
a23     a2^a1      ;Get 3 y sums (to sum conflicts into)
a5      a7+a5      ;Determine conflict: iota2==iota1
s6      v1,a2
a24     a3^a1      ;Determine conflict: iota3==iota1
a15     a5<a29      ;=1 if >=1 more conflicts
s7      v1,a3
a25     a3^a2      ;Determine conflict: iota3==iota2
a6      a5+a15

a16     a1      ;Save iota1
a1      v7,a5      ;Bottom load next iter's iota1
a7      a6<a29      ;=1 if >=2 more conflicts
a17     a2      ;Save iota2
a2      v7,a6      ;Bottom load next iter's iota2
a7      a6+a7
a18     a3      ;Save iota3

a13     a11
s1      a11
a11     a24?a0:a11      ;y1 if iota3==iota1, else 0
a3      v7,a7      ;Bottom load next iter's iota3
a13     a23?a0:a13      ;y1 if iota2==iota1, else 0
s2      a12
a12     a25?a0:a12      ;y2 if iota3==iota2, else 0

s11     a11
a11     v8,a5      ;Bottom load next iter's y1
s13     a13
s12     a12
a12     v8,a6      ;Bottom load next iter's y2

s4,d      s3+s11      ;y3 += (iota3==iota1)? y1 : 0
s3      v8,a7      ;Bottom load next iter's y3
s2,d      s2+s13      ;y2 += (iota2==iota1)? y1 : 0
s4,d      s4+s12      ;y3 += (iota3==iota2)? y2 : 0

s5,d      s5+s1      ;Sum1 += y1
s6,d      s6+s2      ;Sum2 += y2 [+ y1]
s7,d      s7+s4      ;Sum3 += y3 [+ y1] [+ y2]
v1,a16    s5
v1,a17    s6
v1,a18    s7

$UNTIL      a15,Z

noconflict =      *
bn      a26,cnfxloop      ;Branch here if no conflicts
;Repeat if more ix's mapped to same CNFX

a63     a63+a20      ;Restore stack frame

```

```
*  
* End of inlined "indexed partial reduction" algorithm.  
*  
* Update LHS using unique IX mask, M1, and non-allocating gather/scatter.  
* Use ordered (ripple) msyncs if multistreamed.  
*  
    msync a21,v          ;Ordered msync  
    v4      [a32,v2],m1,na      ;Gather TABLE[xdata.index[*]]  
    v5,d  v4+v1,m1  
    [a32,v2] v5,m1,ord,na      ;scatter my updated TABLE values  
    msync a22,v          ;End ordered msync
```

>